Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**ОТЧЁТ №7**

**Дисциплина: Многоагентное моделирование**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Иванов

Направление подготовки: 02.03.03 Математическое обеспечение и администрирование информационных систем

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. А. Миков

Краснодар

2024

**Цель работы**

Запрограммировать карточную игру в «Дурака» 2 на 2, где агенты соревнуются друг против друга разными алгоритмами игры.

**Описание задачи**

Есть 4 агента, поделённые на соперничающий пары и взаимодействующие со средой – карточным столом.

Среда содержит 52 карты, делённые на 7 подмножеств, которые определяют её состояние:

1. У каждого из 4 агентов своё подмножество карт;

2. Открытые карты раунда на столе;

3. Колода «Бито»;

4. Общая колода, из которой добираются карты, с нижней картой-козырем.

Каждый агент видит своё подмножество карт, открытые карты на столе и нижнюю (козырную) карту общей колоды.

Процесс игры выглядит так:

1. Инициализация (перемешивание колоды, выдача карт, определение порядка ходов – всё это выполняется средой);

2. Игра по правилам «подкидного и переводного Дурака» по раундам;

3. Учёт результата.

В каждом раунде у пары определяется ведущий игрок. Ведущие игроки играют друг против друга, напарники же помогают тем, что подкидывают карты.

Ведущим и заходящим игроком первого раунда становится игрок, у которого наименьший номинал карты козырной масти. В соперничающей паре ведущий игрок выбирается случайно.

В ходе игры заходящий игрок кладёт на стол любую из имеющихся у него карт, а отбивающийся игрок (игрок, под которого сделан заход) должен либо побить её, либо взять. Чтобы побить (синоним – покрыть) карту, нужно из имеющихся на руках карт положить на неё более старшую карту такой же масти, либо козыря, если битая карта – не козырь. Если битая карта – козырь, то побить её можно только более старшим козырем. После этого, если у игрока остаётся одна карта, то никто не имеет права атаковать его.

Заходящий игрок или его напарник могут положить (подкинуть, подбросить) ещё одну или несколько карт любой масти, достоинство которых совпадает с достоинством любой из карт, уже участвовавших в данном заходе.

Каждый раунд выглядит так:

1. Заходящий игрок выбирает карту;

2. Если итерация первая, то отбивающийся игрок может перевести ход, если у него есть карта того же достоинства, но другой масти, и добавить её на стол. В таком случае игра продолжается со следующего пункта, но роли изменены;

3. Отбивающий игрок выбирает карту, которой побьёт карту заходящего. Если такой карты нет, то он обязан взять карту;

4. Если отбивающий находит подходящую карту, он бьёт карту заходящего, и тот может подкинуть карту. Если заходящий решает не подкидывать, то он предлагает подкинуть карту своему напарнику;

5. Пункт аналогичен пункту 3, однако если отбивающему нечем крыть новую карту, он забирает все, что лежат на столе (в открытом множестве карт), и раунд заканчивается;

6. Если подкидывать нечего и все карты биты, то раунд заканчивается;

7. Следующий раунд начинает пара, выигравшая предыдущий раунд.

Ротация ведущих игроков в каждой паре остаётся на усмотрение разработчика.

Если в паре у одного игрока закончились карты, то он выходит. Победой считается ситуация, когда у обоих игроков пары закончились карты.

**Задача**

Разработать и описать два алгоритма игры (для каждой пары свой). Провести 1000 экспериментов и сравнить количество побед для разных алгоритмов.

**Описание решения**

Для написания программы был выбран язык Python ввиду его простой реализации ООП.

Для реализации программы было создано 7 файлов: constants.py, Card.py, MainDeck.py, BitoDeck.py, Player.py, Table.py, main.py.

Описание файлов:

constants.py: содержит массив мастей и словарь имён-значений всех карт колоды. Сделано для того, чтобы случайно никак не изменить данные в других классах.

Card.py: класс, реализующий объект Карту. Имеет конструктор с мастью, весом, названием, “козырностью” карты, а также имеет геттеры и сеттеры для безопасного взаимодействия с полями карты.

MainDeck.py: класс Основной колоды, представленной очередью карт с максимальной длиной (52). Имеет соответствующие поля и методы (геттеры, сеттеры и метод “взять карту”).

BitoDeck.py: класс колоды Бито, также реализованный через очередь максимальной длины 52. Имеет геттеры, сеттеры и метод “положить карту”.

Player.py: один из основных классов программы, реализует Игрока. Содержит необходимые поля, такие как «рука», «напарник», «оппонент слева», «оппонент справа» и т.д. Также содержит методы взятия и удаления карты с руки и содержит 4 основных алгоритма игры: случайный ход, случайная защита, экономный ход, экономная защита. Случайный алгоритм предельно прост: игрок выбирает случайную карту и ходит ей на стол (если тот пустой) или формирует список подходящих карт и ходит случайной; защита аналогичная – игрок формирует список подходящих карт и защищается случайной. Экономные алгоритмы работают иначе: экономный ход реализует тактику «лучшее – на конец», т.е. игрок старается избавляться от самых слабых карт самых распространённых мастей в его руке, т.е. бережёт козыри, сильные карты и карты тех мастей, которых осталось мало.

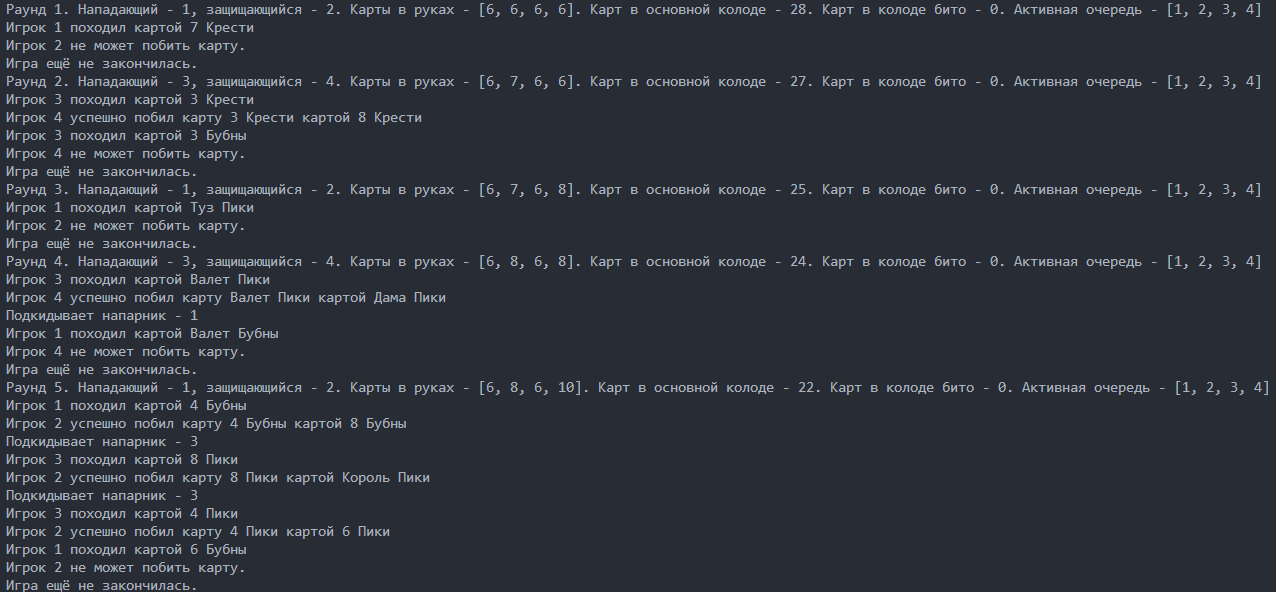
Table.py: основной класс внешней среды – игрального Стола. Стол управляет всей игрой, т.е. создаёт колоду карт, создаёт игроков, настраивает игрокам зависимости (соседей, напарников, организует очередь активных игроков, очередь хода игроков, следит за передачей хода при взятии и битье карт и т.д.). Вся игровая партия сроится следующим образом: игроки ходят по очереди по часовой стрелке – первый на второго, второй на третьего, третий на четвёртого, четвёртый на первого. Игровые пары – первый/третий и второй/четвёртый. Подкидывать карты может только напарник. При взятии карты игрок пропускает ход. Игра продолжается до тех пор, пока из игры не выйдут оба игрока одной пары. Пример игры будет представлен ниже на картинках.

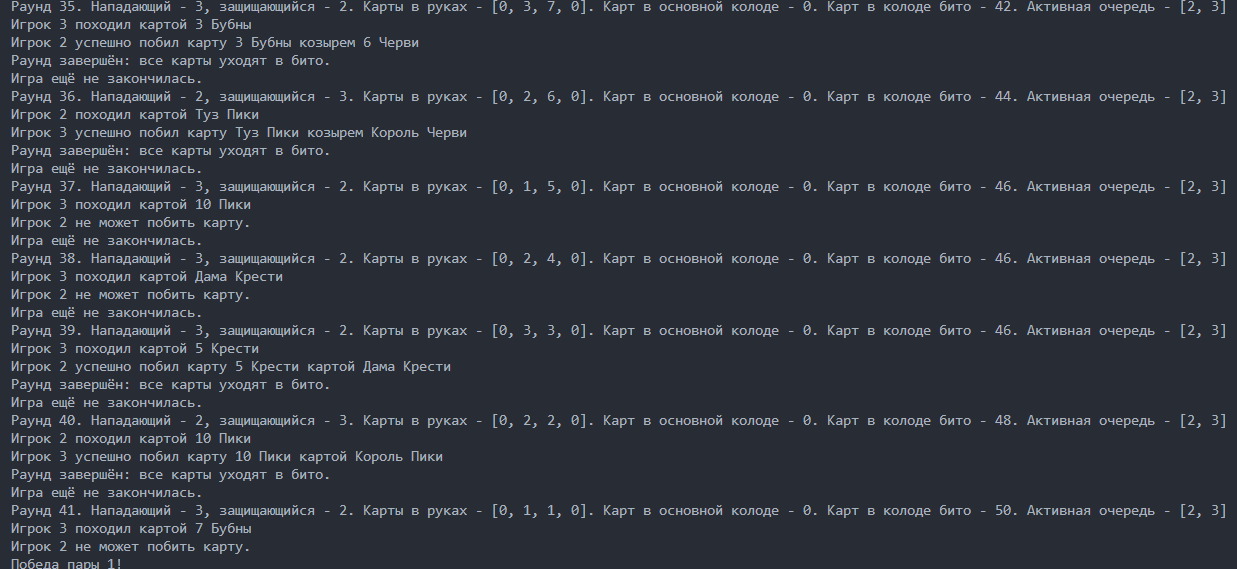
main.py: файл, который запускает основную симуляцию в 1000 игр для сравнения эффективности алгоритмов игры ботов.

**Код программы**

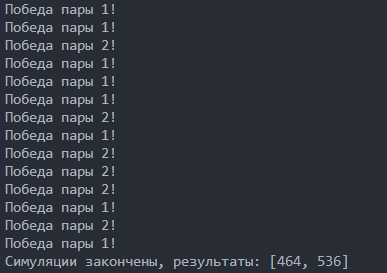
Код программы решено не вставлять в отчёт текстом ввиду его слишком большого объёма (900+ строк). Код будет прикреплён архивом к сообщению с отчётом.

**Пример вывода программы (подробный)**





**Пример вывода программы (1000 игр)**



Первая пара играет со случайным алгоритмом, вторая же с экономным. В среднем, вторая пара с экономным алгоритмом побеждает чаще.